Single-Page Apps and Client-side Routing

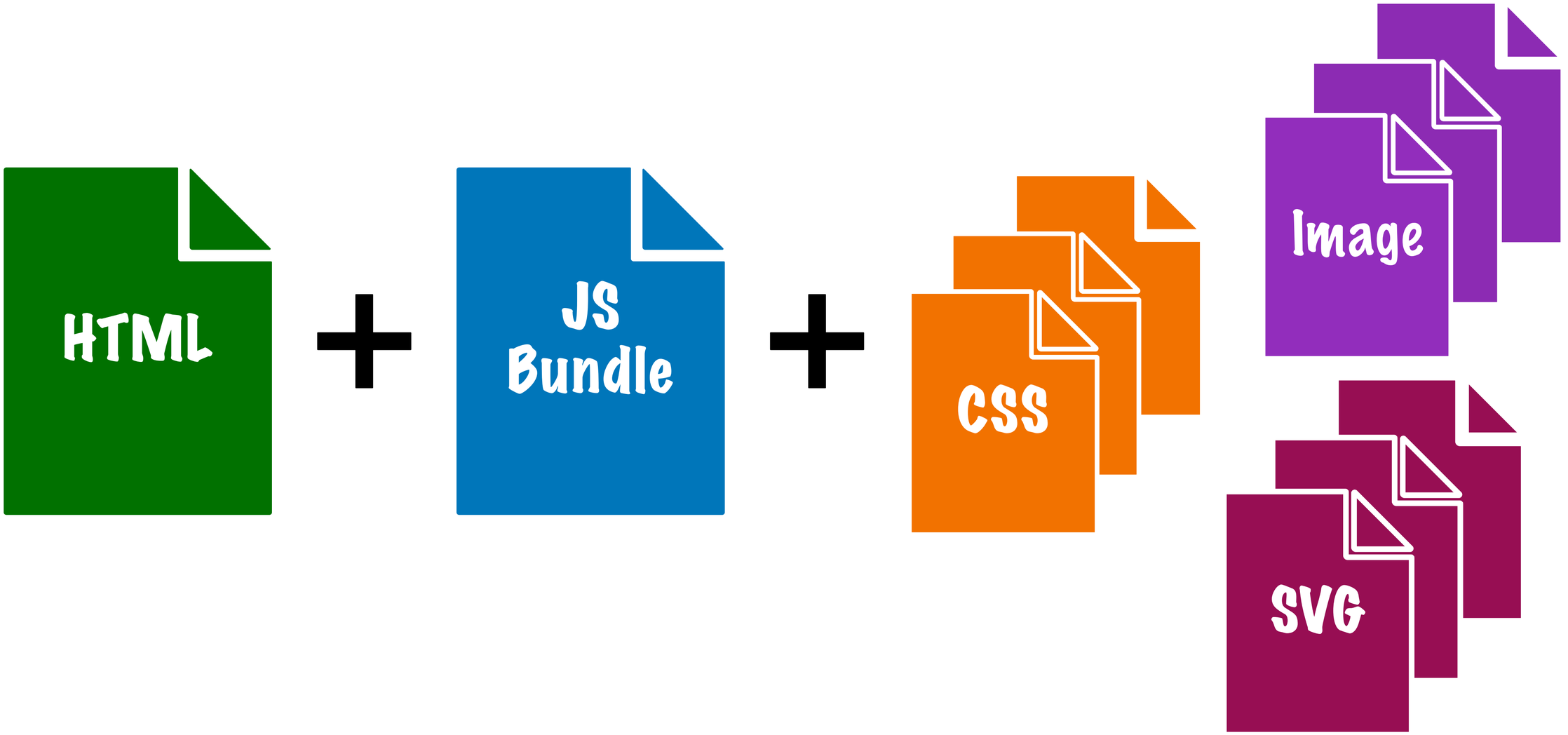
CSC 307 — Week 5 / Day 2

Prof. Ken Kubiak — Spring 2024

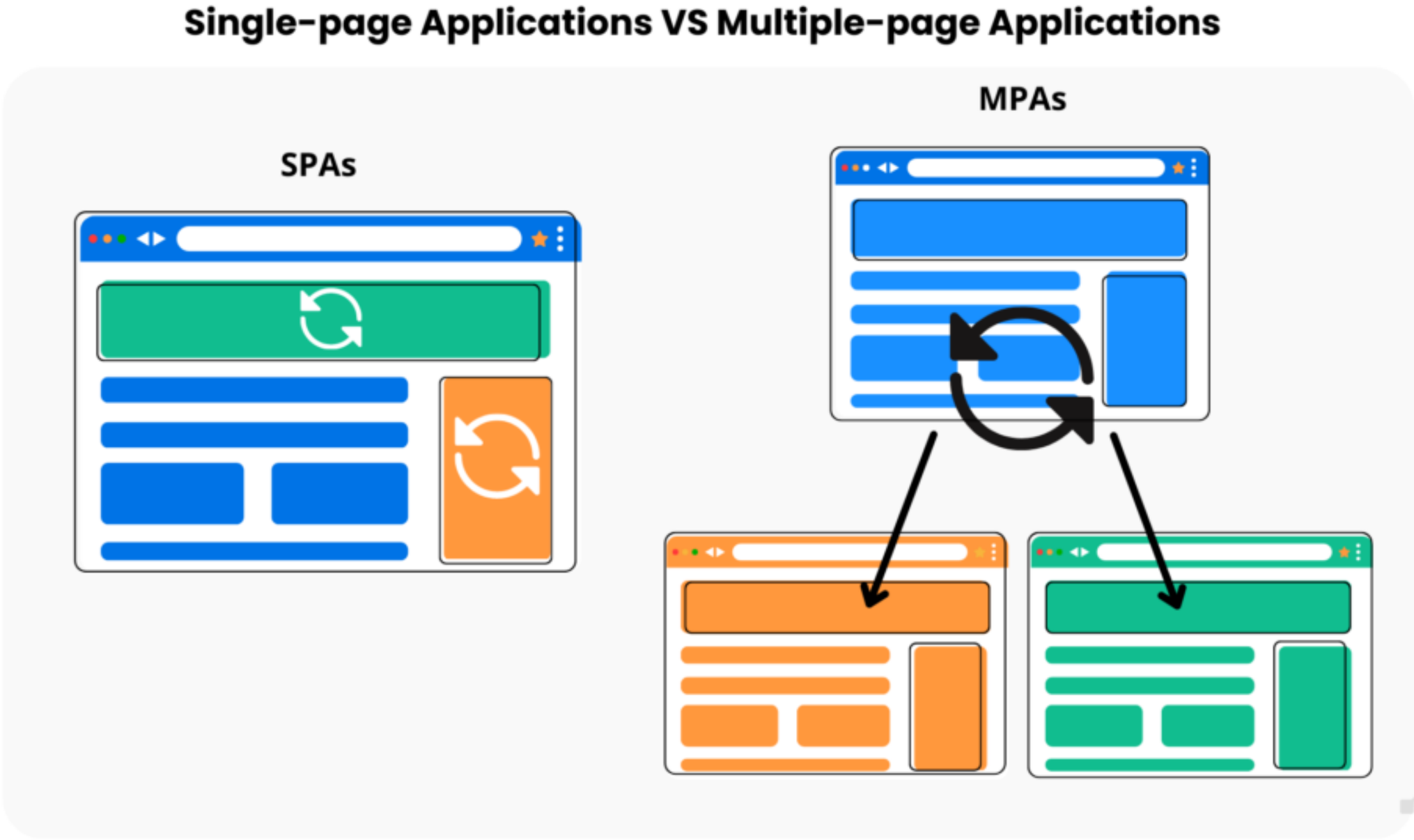
Today's Topics

- ✓ Single-Page Apps
- ✓ Using Vite to Develop & Build SPA
- ✓ Links between "pages" of SPA
- ✓ Client-Side Routing

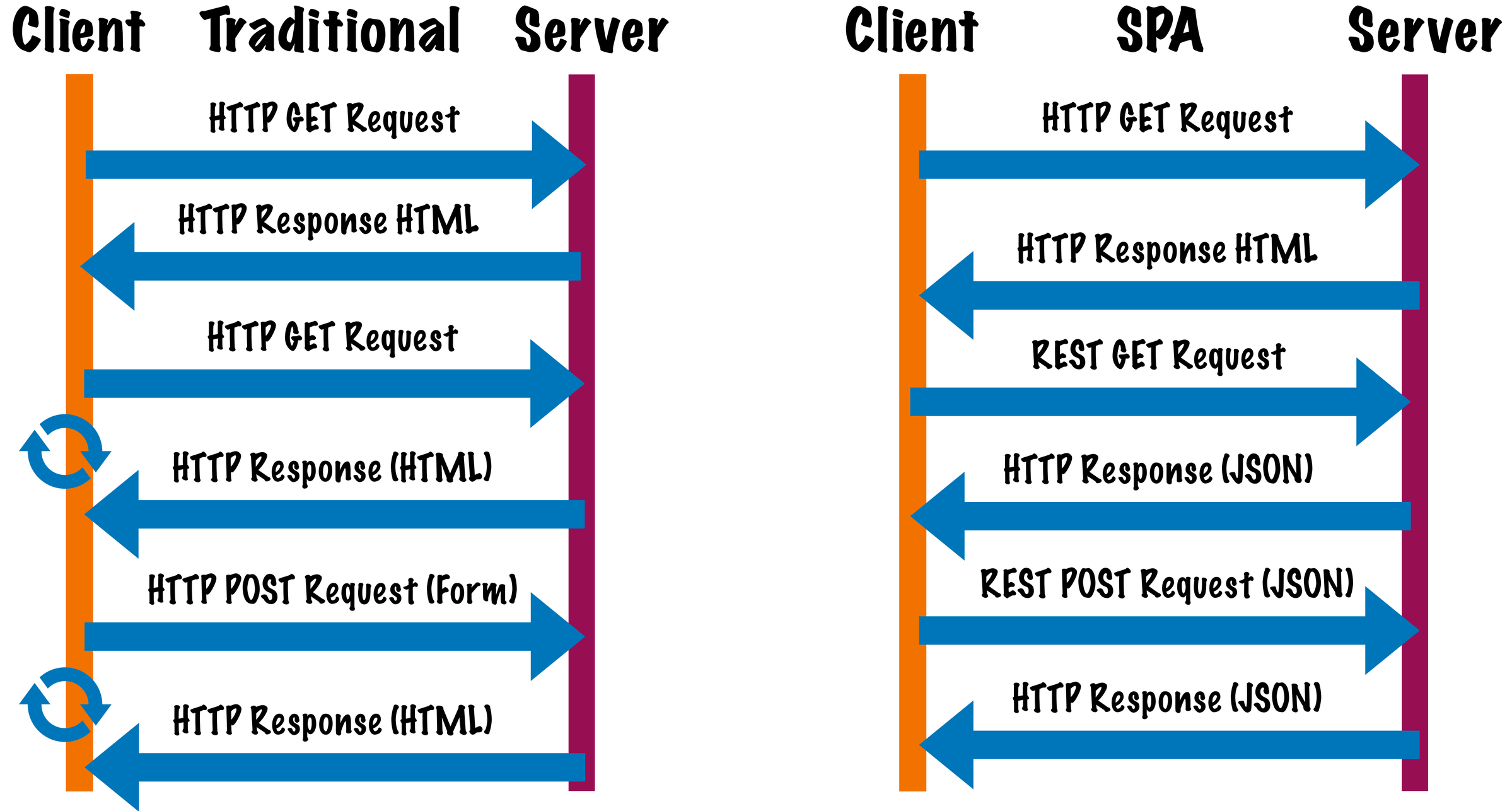
Single-Page means One HTML File



User Experience of SPA



Watch the Request/Response

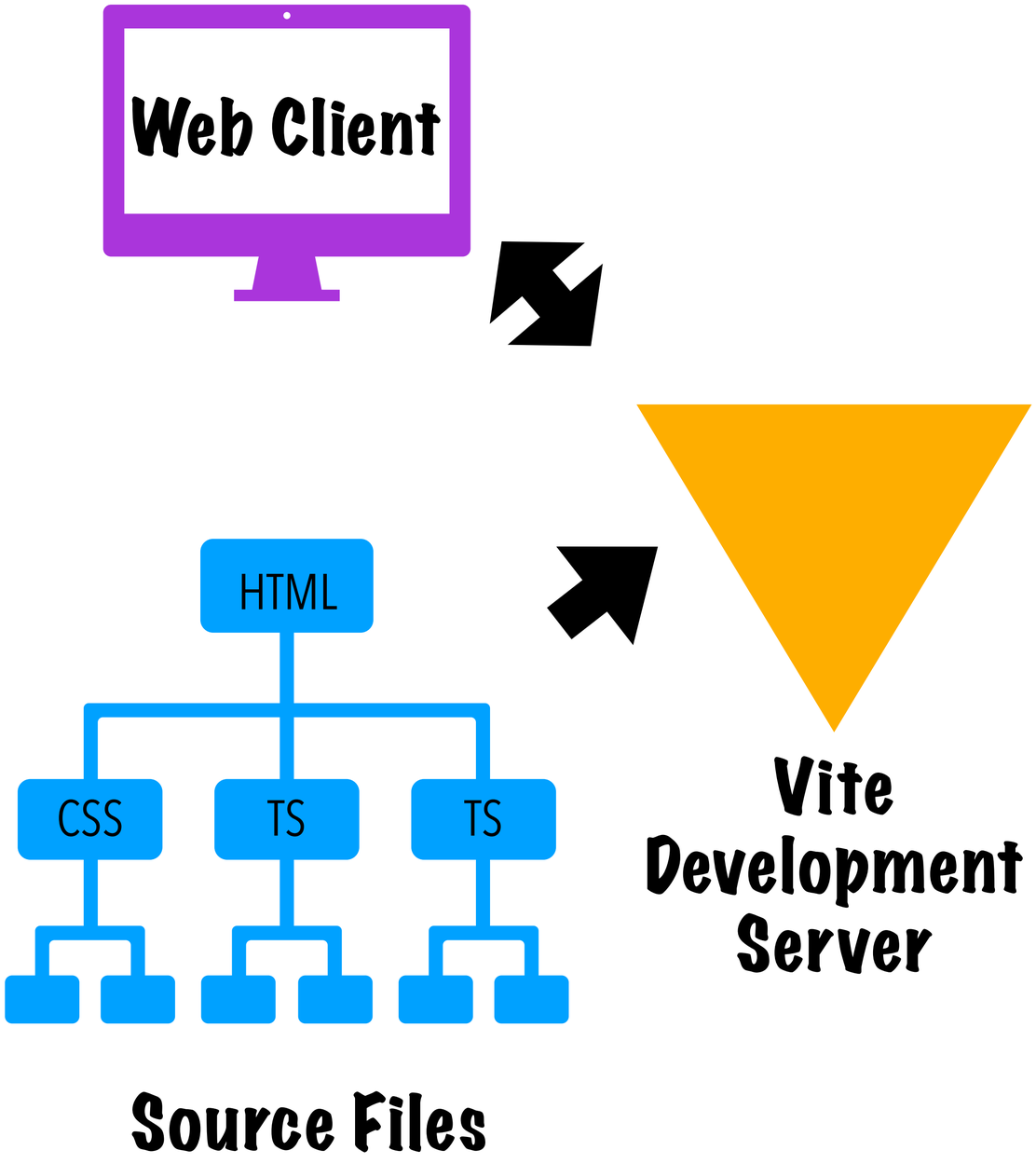


SPA index.html

```
<!doctype html>
<html lang="en">
  <head>
   <meta charset="UTF-8" />
   <meta
      name="viewport"
      content="width=device-width, initial-scale=1.0"
    />
    <title>Vite + React</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

Vite
Development Mode

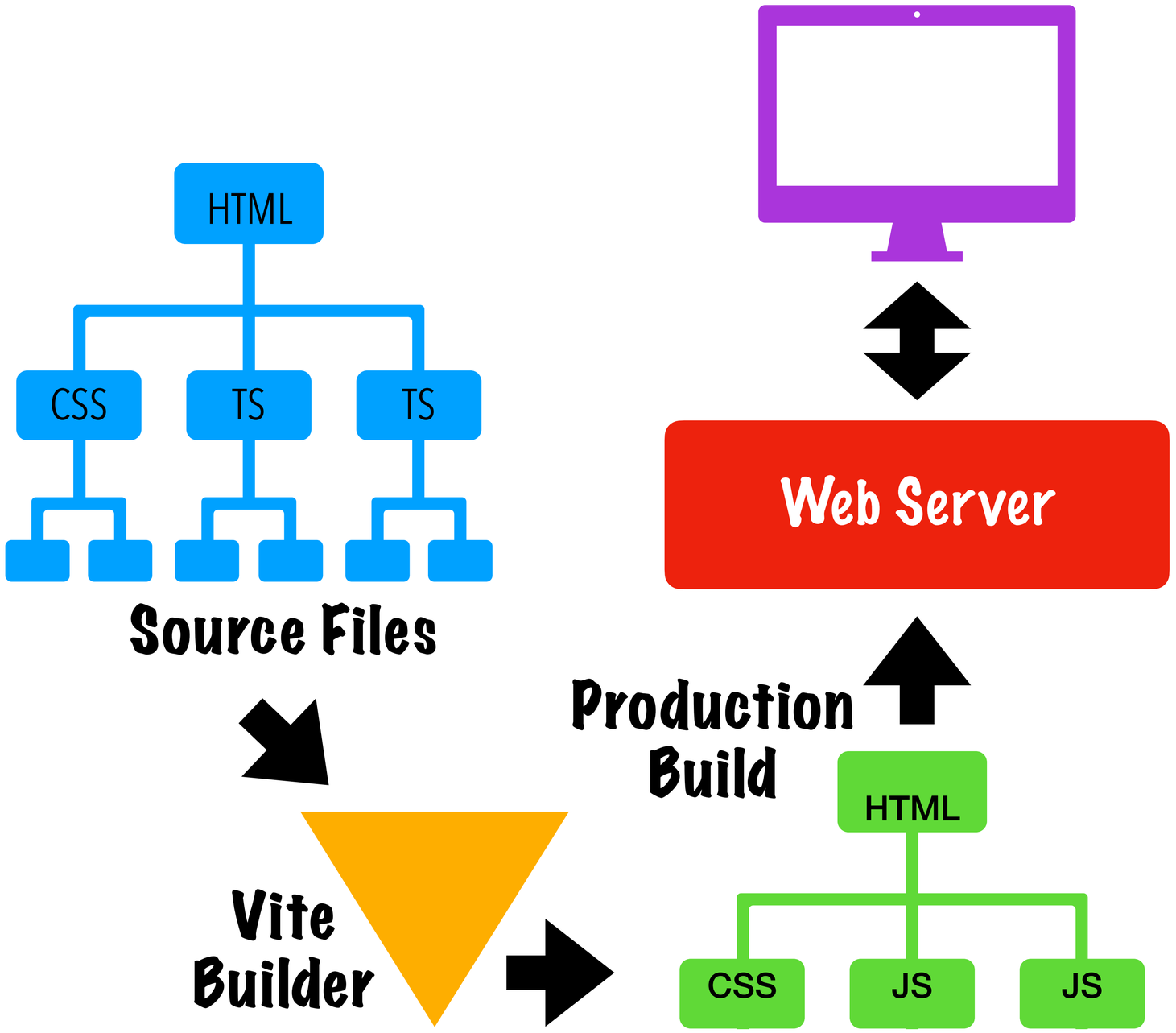
- ✓ Processes all files
  - HTML, CSS, JS, TS, ...
- ✓ Serves bundled versions of the files
- ✓ No build step
- ✓ Hot Module Reload



Vite

Production Mode

- ✓ Processes all files
  - HTML, CSS, JS, TS…
- ✓ Transpiles and Bundles
  - CSS, TS, JS
- ✓ Processes HTML:
  - `<link>`, `<style>`, `<script>`
- ✓ Generates a Production Build folder
- ✓ Served by standard HTTP server



Vite Build creates a dist directory

```
npx vite build
```

```
react-frontend
├── index.html
├── public
│   ├── icons
│   ├── images
│   └── styles
└── src
    ├── components
    ├── models
    ├── styles
    └── views
```


```
react-frontend/dist
├── index.html
├── assets
│   └── app-150b8350.js
├── icons
├── images
└── styles
```

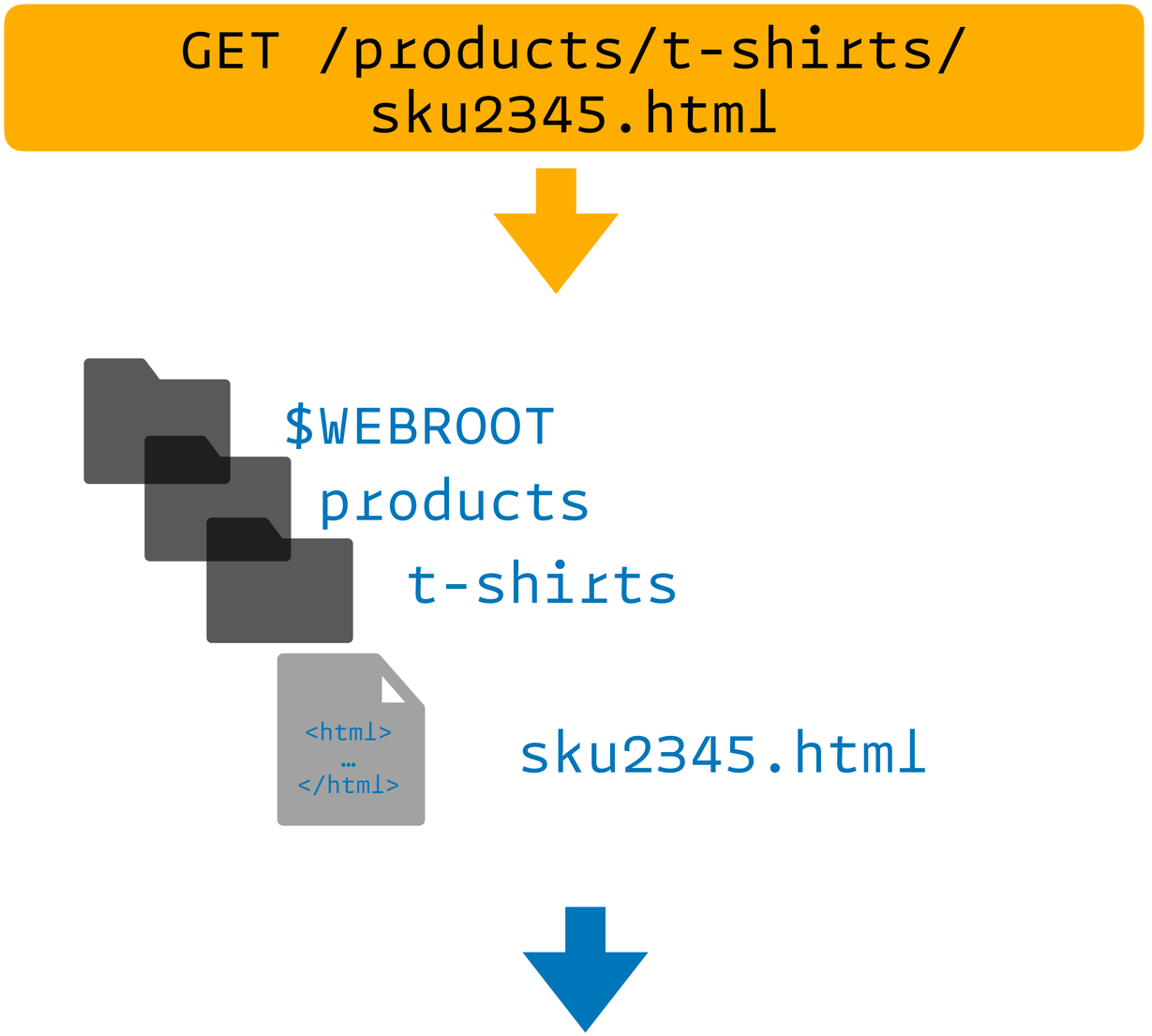
What about links?

Routing on the Server

Static Web Server

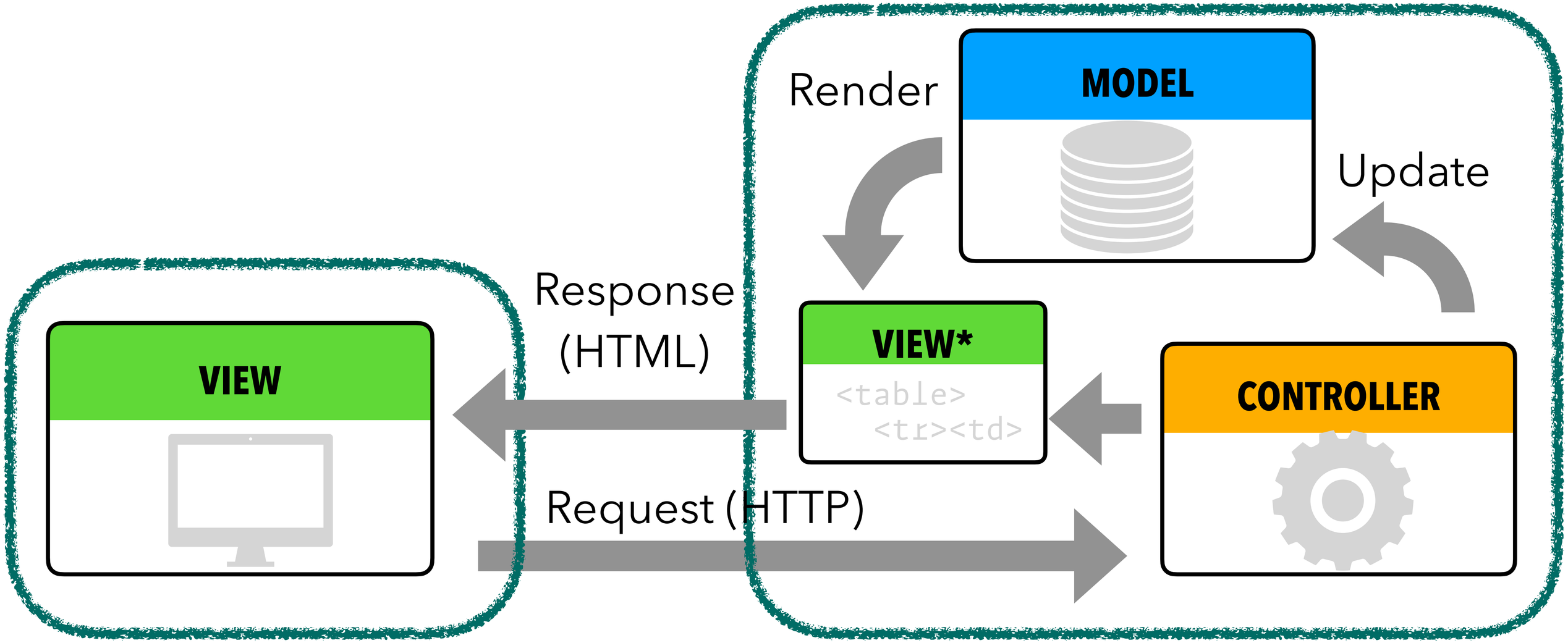
- ✓ URL = Uniform **Resource** Locators
- ✓ Routing:

  Locate file using URL as pathname



MVC Architecture

Traditional Server-rendered Web “Pages”

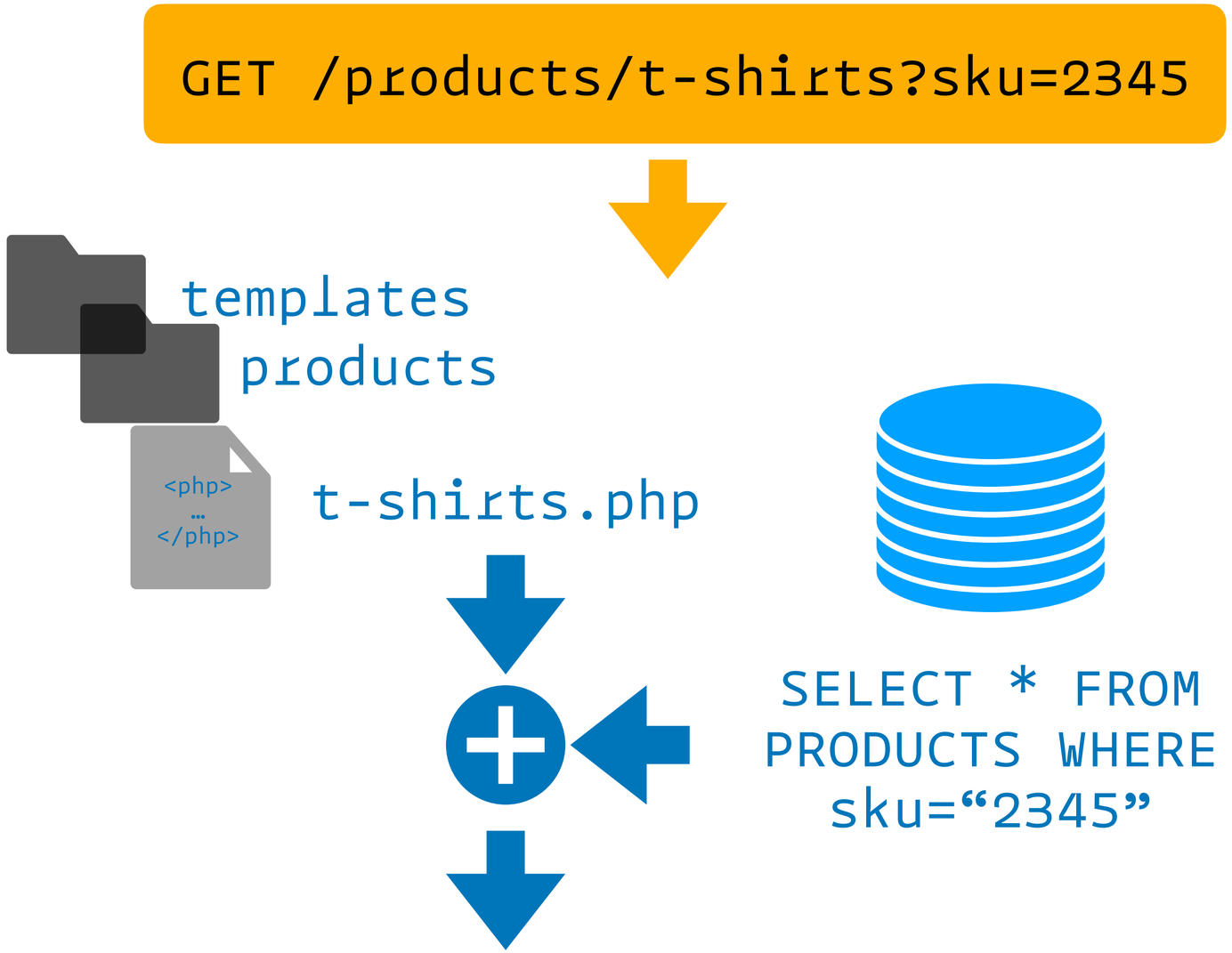


Routing on the Server

Dynamically Rendered Pages

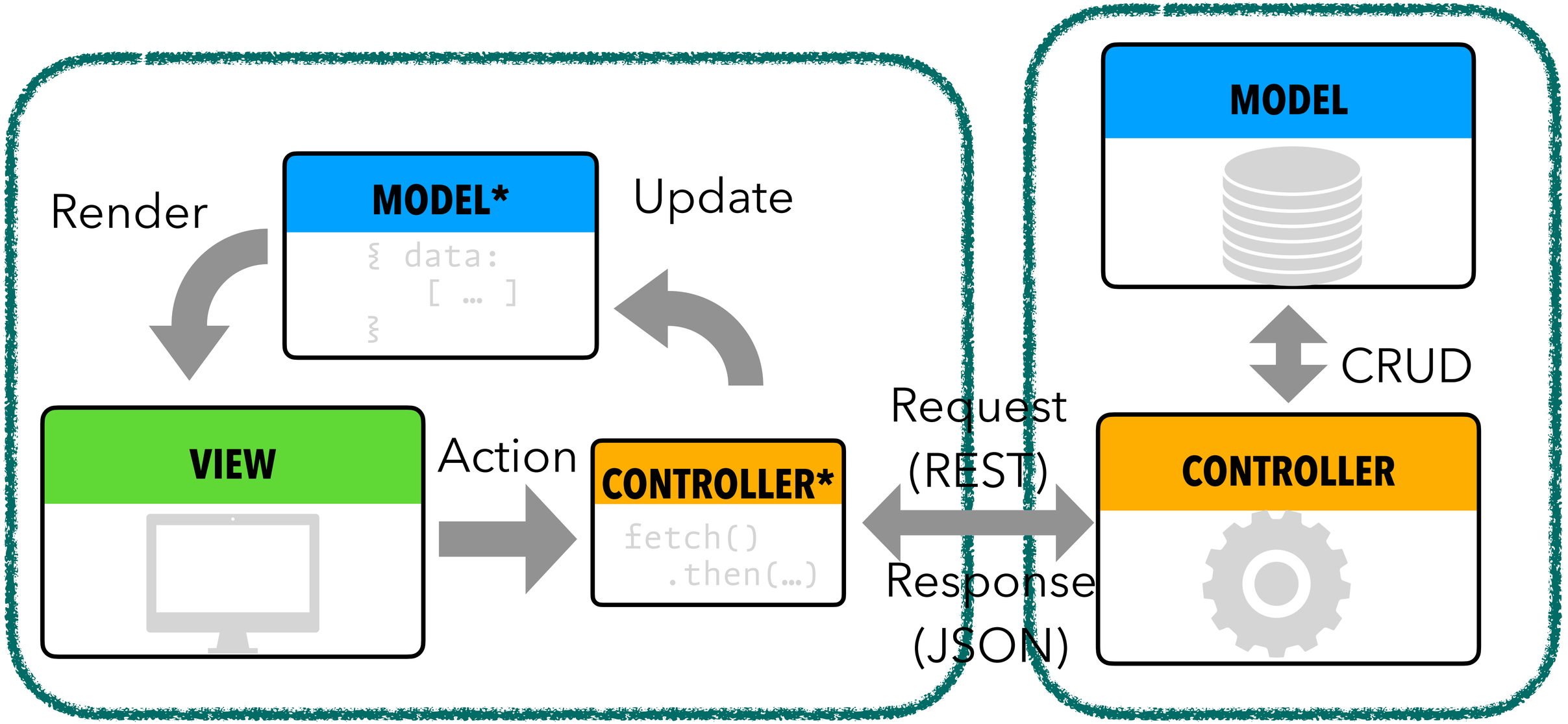
✓Routing:

Locate a template file, and use parameters from URL to query DB

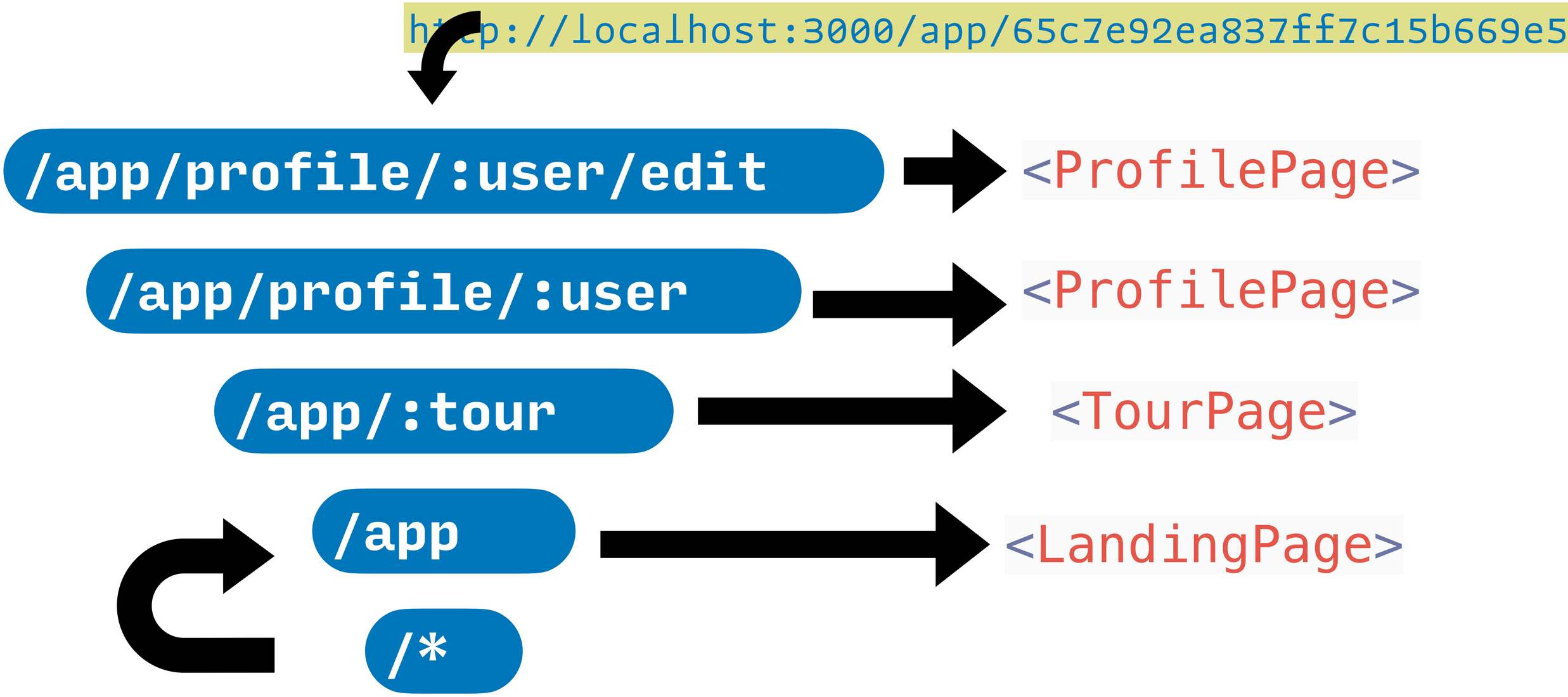


MV* or MVU Architecture

Client-rendered Web Application



Client-side Routing



Routing on the Client

Single-Page Apps (SPA)

✓Routing:

- Identify a component to render
- Pass parameters to API to load data

```
GET /app/products/t-shirts/2345
```

```
<Route path="/products">
  <Route path="t-shirts/:sku"
      component={TShirtPage}/>
  …
</Route>
```

```
<TShirtPage sku="2345"/>
```

Using React Router

```
import { BrowserRouter, Routes, Route } from 'react-router-dom';

...

<BrowserRouter>
  <Routes>
    <Route path="about" element={<AboutPage />} />
    <Route path="products"element={<ProductsPage />}>
     <Route path="t-shirts/:sku"
      element={<TShirtPage />}
     />
     ...
    </Route>
  </Routes>
</BrowserRouter>
```

URL Parameters

```
import { useParams } from 'react-router-dom';

function TShirtPage () {
  let params = useParams();

  fetch(`/api/products/${params.sku}`)
    .then( ... );

  return (
    <article>
      ...
    </article>
  );
}
```

Linking to other “Pages”

Use

<Link to=“…

>

instead of

<a href=“…”

```
import { Link } from "react-router-dom";

function ProductsPage({ products }) {
  return (
    <div>
      <h1>Products</h1>
      <ul>
        {products.map((prod) => (
          <li key={prod.sku}>
            <Link to={prod.sku}>{prod.name}</Link>
          </li>
        ))}
      </ul>
    </div>
  );
}
```

Programmatic Navigation

```
import { useNavigate } from "react-router-dom";

const Home = () => {
  const navigate = useNavigate();

  return (
    <>
      <h1>Home</h1>
      <button onClick={() => navigate("/about")}>
        About
      </button>
    </>
  );
};
```

Assignments

- ✓ Quiz #4: Vision, Stories, Requirements, and SDLC
  - Available Today @ 5
  - Due **This Friday** at 11:59pm
- ✓ Team Exercise #2: UI Storyboards and Clickable Prototype
  - Assigned Friday
  - Due **Next Friday** at 11:59pm